

Accelerating Linux/Windows File Systems by Predicting Access Frequency

Frank Wang¹, C. Liao¹, N. Helian², Chris Thompson¹, S. Wu¹, Y. Deng¹, V. Khare¹ & A. Parker³

¹Centre for Grid Computing, Cambridge-Cranfield High Performance Computing Facility (CCHPCF), MK43 0AL, UK

²Department of Communication, Mathematics and Computer, London Metropolitan University, London, N7 8DB, UK

³Cambridge e-Science Centre, Cavendish Laboratory, Cambridge University, Cambridge, CB3 0HE, UK

Corresponding contact: frankwang@ieee.org

Abstract

Network File System (NFS, de facto in Linux) or Common Internet File System (CIFS, de facto in Windows) is used for applications to move files across networks/grids. We prove that once a file is created with a set of attributes, such as name, type, permission mode, owner and owner group, its future access frequency is predictable. A decision-tree-based predictive model is established with an integrated fuzzy logic facility to further calculate the degree to which a file may be frequently accessed. By consulting with the rules generated from the predictive model over diverse real-system NFS traces, it can predict a newly created file's future access frequency with a sufficient accuracy (greater than 90%).

Keywords: file access frequency, decision tree, classification, flash memory, caching, e-Science

1. Introduction

The file system designers have suggested that the knowledge behind the users' behaviors and file attributes such as file access pattern, size, and name can conduct the improvement and the design of file caching policies and disk layout. The recent researches [1] [2] have proved that file size, lifespan and access mode (write, read, lookup) are predictable to improve file pre-fetching and caching. Unfortunately, no work has been reported in predicting file access

frequency, which is believed one of the most useful factors for file caching and disk layout [3]. In this paper, we show that applications already give useful hints to a file system, in the form of file attributes at creation time, and that the file system can successfully predict its future access frequencies from these hints (Fig.1). The file access frequency in our work is defined as the frequency of various file operations, which include read, write, execute, lookup and rename.

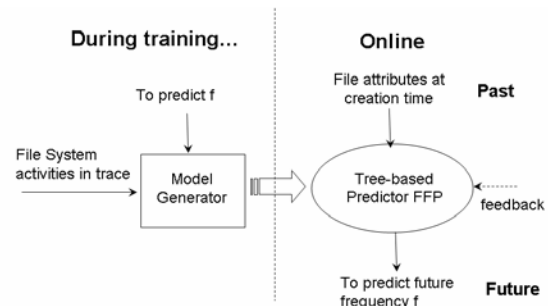


Fig.1 Decision tree-based predictor FFP is a bridge linking the past (attributes at creation) and the future (frequency). During the training period a predictor for file frequency is constructed from observations of file system activity in trace. The file system can then use this model to predict the frequency of newly-created files.

In this paper, we first prove that the future file access frequency is predictable and strongly related to the file attributes at creation time. Then, we construct a decision tree based predictive model to infer a new file's future frequency class

(high, low). Moreover, we define a fuzzy frequency coefficient to calculate the degree to which a file may be frequently accessed to maintain its best performance.

The rest of the paper is organized as follows: Section 2 discusses the related works. Section 3 describes and analyses three NFS traces used in our analysis. Section 4 provides the statistical evidence of file access frequency prediction. Section 5 presents FFP (File Frequency Predictor). Section 6 validates FFP. Section 7 concludes.

2. Related works

Since the gap of processing speed between I/O and CPU performance increases every few years, many efforts have been devoting to address it through optimizing various aspects of file system. One of those efforts is to reduce the number of disk requests by improving the caching efficiency. Another solution is to reduce the disk requesting time through re-organizing the disk layout of the relative files' blocks.

One of such heuristics is the Fast File System (FFS) [4]. It was designed to handle files in different manners regarding to their size. It places small files on disk so that they are near their metadata. Furthermore, it assumes that files in the same directory are most likely accessed together in a short period.

In addition to file size, other properties such as access type: write-mostly or read-mostly, have been confirmed useful to design various file system policies. For example, Log-structured File System (LFS) [5] proposed by John K. Ousterhout and Fred Douglass, treats the disk as a circular log and writes sequentially to the head of the log. The design of log-structured file systems is based on the hypothesis that write latency is the bottle neck of modern file systems. As a result, a Hybrid file system structure designed against the

nature of read and write has been found useful to form high performance file system.

Besides above heuristics, it is noticed that file access frequency is also useful. A data file that is currently being heavily accessed by users is a high frequency file. In a Hierarchical Storage Management system (HSM), file I/O may be constantly monitored in order to migrate the high frequency files to the fastest storage devices or to internal memory for better performance. HSM was first implemented by IBM on their mainframe computers [6] to reduce the cost of data storage, and to simplify the retrieval of data from slower media. The user would not need to know where the data was stored and how to get it back, the computer would retrieve the data automatically. The only difference to the user was the speed at which data is returned.

Based on our investigation, the problem is that all of those heuristics are embedded into file system it selves, which means if the heuristics are wrong or dynamic the system performance will degrade [2]. In addition, file system workloads may change over time, the designed policies may take critical time to adapt to it. In previous work, a solution to this problem is that enable applications to supply hints to the file systems [5]. Unfortunately, this solution requires the modification of applications, which hasn't been widely deployed.

In fact, the workloads, user behaviors and applications already provide very rich hints to conduct file system behaviors. An attributes based file properties prediction method has been introduced recently [2]. It utilizes machine learning approach to analysis the previous file system behaviors and then makes decisions advising file systems the estimated future properties of a new file, such as size, life span and even write-mostly or read-mostly.

However, to the best of our knowledge, there is no reported investigation against file access frequency prediction so far. In this work, we will prove that once a file is created with a set of attributes, such as name, type, permission mode, owner and owner group, its future access frequency is predictable. A decision tree-based predictive model will then be established. The predicted frequency information will be used to decide what files to keep in a flash memory. Hence the work is believed to be timely and novel considering file access frequency is useful to realize most of the performance gains.

3. The Traces and their skews

The Traces we used were provided by Harvard University [6], which were taken from three servers that have different users and behaviors.

“**DEAS03** traces a Network Appliance Filer that serves the home directories for professors, graduate students, and staff of the Harvard University Division of Engineering and Applied Sciences. This trace captures a mix of research and development, administrative, and email traffic. The DEAS03 trace begins at midnight on 2/17/2003 and ends on 3/2/2003.” [2]

“**EECS03** traces a Network Appliance Filer that serves the home directories for some of the professor, graduate students, and staff of the electrical Engineering and Computer Science department of the Harvard University Division of Engineering and Applied sciences. This trace captures the canonical engineering workstation workload. The EECS03 trace begins at midnight on 2/17/2003 and ends on 3/2/2003.” [2]

“**CAMPUS** traces one of 14 file systems that hold home directories for the Harvard College and Harvard Graduate School of Arts and Sciences (GSAS) students and staff. The CAMPUS

workload is almost entirely email. The CAMPUS trace begins at midnight 10/15/2001 and ends on 10/28/2003.” [2]

For the privacy issue, some information of these three traces was encoded by using the method described in [7]. The encoded traces contain anonymized UIDs, GIDs, and IP address remapped to the new values. There are no information lost among those identifiers and other variables. During the anonymization UIDs, GIDs, and host IP numbers are simply remapped to new values, so no information is lost about the relationship between these identifiers and other variables in the data. All three traces were collected with nfsdump.

Host	read	write	lookup	getattr	access
DEAS03	48.7%	15.7%	3.4%	29.2%	1.4%
EECS03	24.3%	12.3%	27.0%	3.2%	20.0%
CAMPUS	64.5%	21.3%	5.8%	2.3%	2.9%

Table 1 The average percentage of read, write, lookup, getattr, and access operations for a fourteen day trace.

Table 1 gives a summary of the average hourly operation counts and mixes for the workloads captured in the traces. These show that there are differences between these workloads, at least in terms of the operation mix. CAMPUS is dominated by reads and more than 85% of the operations are either reads or writes. DEAS03 has proportionally fewer reads and writes and more metadata requests (getattr, lookup, and access) than CAMPUS, but reads are still the most common operation. On EECS03, meta-data operations comprise the majority of the workload.

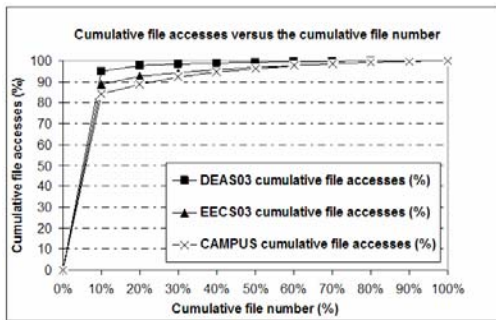


Fig.2, The cumulative file accesses versus the cumulative file number, both expressed as percentages. It was found that more than eighty percent of the file accesses are probably going to less than ten percent of the files.

In Fig.2, we plot the cumulative file accesses versus the cumulative file number, both expressed as percentages. The file accesses in our work include read, write, execute, lookup and rename operations. The curve of DEAS03 in Fig.2 shows 94.9% of the file accesses going to 10% of the files. This result with curve of EECS03 and CAMPUS become less skewed. In EECS03 88.9% of the file accesses are going to less than 10% of the total files whereas in CAMPUS 84.1% of the file accesses are going to less than 10% of the total files. CAMPUS is similar to EECS03, but contains a lot of sequential access patterns. In summary, file accesses are highly skewed and experience shows that only a small fraction of files in a file system is actively and frequently used. As also shown in Fig.2, the exact shape of the file request distribution varies from workload to workload.

4. Statistical Evidence of Association

To identify the associations between the create-time attributes of a file and its future access frequency, we first scan the traces to obtain the previous knowledge and calculate the file access frequency. In these 3 NFS traces, the attribute “fid” uniquely identifies a file; therefore, we can record the access frequency of a file by accumulating the number of occurrences of a

certain “fid”. Once we obtain the access frequency of a file in a short period, say peak hours (10am-1pm), we are able to measure the statistical association between each attribute and the relative file access frequency.

The fact is that some of the associations are clear and easily to be identified. For example, the lecturers regularly update his/her lecture notes in the file server. Consequently, in the next short period, the files created with UID=”professor A” will be accessed frequently. In contrast, for example, if a student submits his/her coursework to the file server, in the next short period, the files created with UID=”student B” will not hold a good probability of being accessed as frequent as previous one regarding to the popularity reason.

Similarly to this, file mode can also provide useful information to file access frequency. According to the observation in [2], any files with a mode of 777 in DEAS03 trace is likely to live for less than one second and contain zero bytes of data. It suggests that if a file is created with a mode that makes it readable and writeable by any user on the system is actually never read or written by anyone. For example, lock files (files that are used as a semaphore for interprocess communication) usually exists only for indicating that a process desires exclusive access to a particular file. Therefore, the information hidden behind file access mode can also be used to evaluate for file access frequency.

To statistically confirm the degree that how strong an attribute of a file relates to the file’s access frequency, we use Chi-square test [8] to roughly analysis the association between high access frequency files and each attributes. Chi-square test lets us know the degree of confidence we can have in accepting or rejecting a hypothesis. Concretely, the hypothesis of our Chi-square test is whether or not a high access frequency file’s

attributes and its access frequency class (high) are associated enough. Given access frequency boundaries are 599, 19, and 17 respectively regarding to the 10% top frequent accessed files out of the total files, the Fig.3, Fig.4 and Fig.5 show the confidence of the high access frequency class supported by each attribute over three NFS traces.

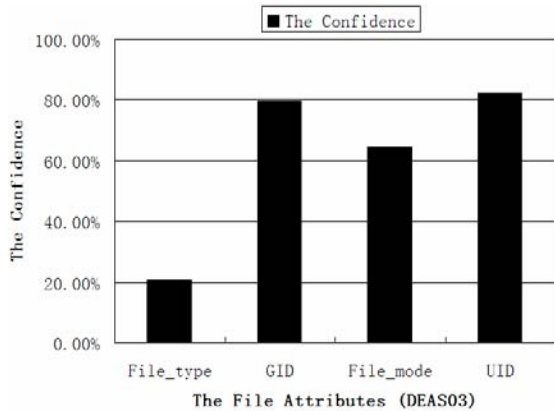


Fig.3. The confidence of file access frequency of attributes (DEAS03 18/2/2003).

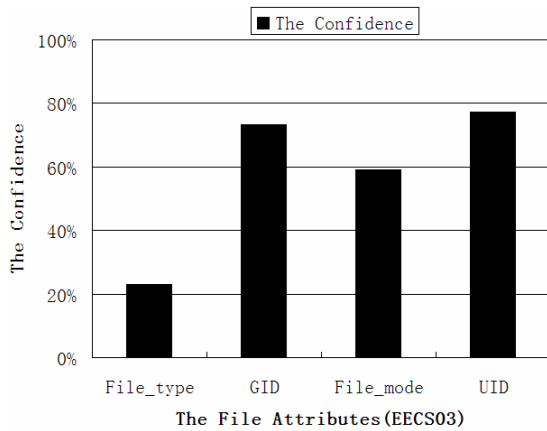


Fig.4 The confidence of file access frequency of attributes (EECS03 18/2/2003).

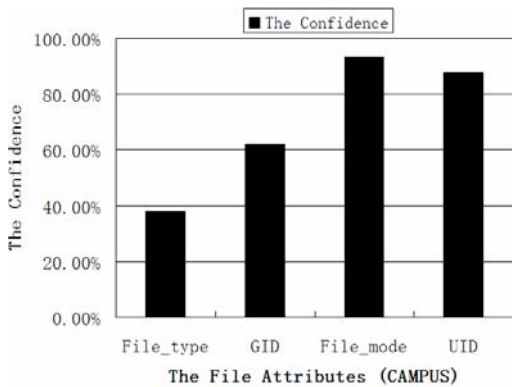


Fig.5 The confidence of file access frequency of attributes (CAMPUS 16/10/2001).

The figures above reveal the confidence of file access frequency supported by the diverse attributes. It can be seen from the figures that the confidence values supported by each attribute are different in each trace. However, some attributes such as UID always show great confidence to the file access frequency, which is one of the representative cases we have discussed at the beginning of this section.

As the Chi-square test is a rough statistical analysis of the confidence of file access frequency, there is a need to precisely evaluate the degree of confidence supported by each attributes. Here we borrow the concept from information theory, Information Gain, to discover how much contribution an attribute can provide to the confidence of file access frequency prediction. This approach is also used to build a more accurate statistical analysis model, decision tree. In the next section, we discuss our decision-tree based predictive model.

5. File Access Frequency Classifier (FFP) Construction

The Chi-square test in previous section has provided us the evidence that files attributes are strongly related to files access frequency. It suggests us that if we know a file's attributes, we then can predict a file's frequency class (high or low). Apart from those representative attributes mentioned above, some other attributes such as the file's creation time, file name and suffix may also support file access frequency. In this paper, we focus on 4 attributes: file mode, file type, user ID, and group ID.

In UNIX file system, user ID and group ID identify whom the file belongs to and its group. Correspondingly, file mode claims the permission title in three separate parts. For example, if a file

contains a user ID “john”, group ID “G1” and the file mode “-rw-r-r”, then it claims that “john” is the owner of this file and has the group “G1”; in the file mode, first set value “rw” denotes that “john” can read and write this file, second set value “r” claims that group users belongs to “G1” can only read this file; the third set value denotes other users can only read this file.

In addition, there are 4 main file types in UNIX file system: ordinary files, directory files, device files, and link files. Each file type represents different file contents, an ordinary file may contain texts, programs, or other data; a directory file contains the information stating that what files in this directory, how large they are, when were they last modified, etc. ; the device files represents physical devices; the link files are alternatives of same physical files. Here we don't discuss detailed concepts of UNIX file types.

To accurately evaluate the confidence given by files' attributes, we use decision tree as our predictive model. We first obtain training sample to build the predictive model, and then validate the model with new files to check whether or not the predictions are accurate. There are many learning algorithm for data classification, the most popular one is Iterative Dichotomiser 3 (ID3) decision tree [7], which utilizes the information gain to evaluate the confidence supported by an attribute. Decision tree requires categorical attributes and fixed classes. For the first requirement, since file attributes such as file type, file mode, UID, GID are all symbolic represented with no inherent ordering, therefore our classification problem satisfies this requirement. For the second requirement, we can define a caching performance-based frequency boundary to distinguish low frequency and high frequency.

The files we collected are from three NFS traces: DEAS03, EECS03 and CAMPUS. Through

scanning the traces, we can capture the file attributes (file type, file mode, UID, GID) and its access frequency by accumulating the occurrences of the file requests. The following table shows a sample taken from DEAS03, the frequency is regarding to one hour (03/02/2003, 00:00-01:00).

Suppose we define 599 is the boundary between low frequency and high frequency of the sample from DEAS03, we can obtain the frequency class in accordance of their real access frequency.

File type	File mode	UID	GID	Access frequency
1	180	18b34	6	7964 (high)
1	180	18ad4	6	242 (low)
1	180	18aa2	6	299 (low)
1	180	18c09	18b3b	567 (low)
2	180	18c09	18b3b	21546 (high)
2	7ff	0	0	6689 (high)
2	5c0	18a89	18a89	7145 (high)
1	5c9	18aa2	18ac2	54 (low)
2	1c0	18aee	18ad5	3657 (high)

Table 2. Training sample from DEAS03.

By a given training sample, ID3 algorithm takes all attributes and count their information gain, then choose the attribute for which information gain is maximum, at last make nodes containing that attribute. From the observation of ID3 algorithm, we noticed that from the top of the tree to the bottom of the tree, the upper level nodes hold larger information gain, which means the upper level nodes that belong to a certain attribute make more contribution for classifying the access frequency. The Fig.6 is the ID3 decision tree built regarding to the above training sample.

It can be seen from Fig.6 that begin from the Root, each branch represents a classification rule and its class label. In Fig.6, not all rules are populated in the example tree so that the example decision tree is more readable.

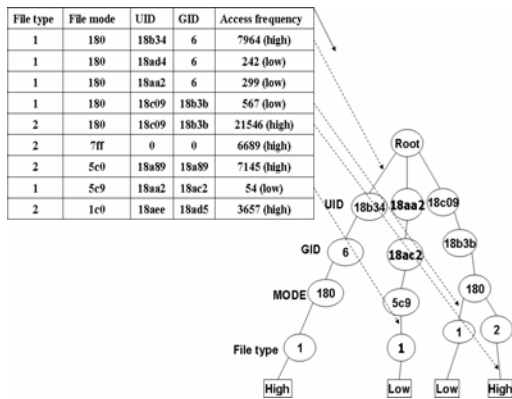


Fig.6 ID3 decision tree of the training sample.

The classifier we built in this section is based on the decision tree classification algorithm ID3. However, the training sample we used in the example for building the classifier is very small. Generally, more training data used for building the classifier more precise the result reaches the reality. In next section, we will discuss the validation of FFP.

6. The validation of FFP

For other time-irrelevant classification problems, training samples and test samples are normally split from same workloads to guarantee both samples have same data characteristics. However, in our problem, those traces are obtained in an uninterrupted time series over several days. Thus, each hour's trace may differ from each other in terms of file access frequency. It is impossible to split the whole time series workload into first half and second half to build classifier and validate it. As we can imagine, in peak hours, file access frequency could be higher than rest of hours.

To evaluate the accuracy of FFP over the time series traces, we built the classifiers based on the peak hours' traces (10am-1pm) on Monday (02/17/2003) as the training sample. Then we predict the files created during the peak hours in the following days to check whether or not the precision is acceptable. The following tables show the accuracy of the predictions during the peak

hours on Tuesday, Wednesday and Thursday.

Frequency class	DEAS03	EECS03	CAMPUS
high	90.1%	92.5%	93.3%
low	98.2%	91.2%	90.6%

Table 3 Prediction accuracy of FFP on Tuesday.

Frequency class	DEAS03	EECS03	CAMPUS
high	88.4%	88.0%	90.1%
low	93.6%	89.6%	87.7%

Table 4 Prediction accuracy of FFP on Wednesday.

Frequency class	DEAS03	EECS03	CAMPUS
high	82.6%	87.7%	84.1%
low	88.6%	85.3%	82.2%

Table 5 Prediction accuracy of FFP on Thursday.

The validation results show that the accuracy of the classifier decreases after few days. This is resulted by the slightly changed user behaviors and files' life span. For instance, some lecture notes might be accessed frequently in the next few days after the corresponding lectures were given. However, those lecture notes' access frequency may be no longer frequent once some new lectures are given and their relative lecture notes are created. Hence, there is a need to build a new predictive model to adapt the file system change. The possible way is to rebuild the predictive model when the precision of the model becomes unacceptable. In next section, we discuss some possible scenarios to apply file access frequency prediction.

7. Conclusion

In this paper, we describe data mining technology embedded in a commercial product (An Evolutionary Storage System integrating file frequency predictor, FFP). We proved that file attributes are strongly related to its access frequency. We have also presented a method of predicting file access frequency. The

decision-tree-based predictor FFP is supposed to be a bridge linking the past (attributes at creation) and the future (frequency). During the training period a predictor for file frequency is constructed from observations of file system activity in trace. The file system can then use this model to predict the frequency of newly-created files with an accuracy of more than 90%. The frequency information could be used to decide what files to keep in a flash memory. The trace-driven experimental results indicate that the performance speedup due to the prediction-enabled optimization is 2 - 4.

Acknowledgements

We thank Jonathan Ledlie and other SOS [9] project researchers for providing us the simulation traces and the heuristics used in our design.

Reference:

- [1] T. M. Kroeger and D. D. E. Long, "The Case for Efficient File Access Pattern Modeling," in *Proceedings of the Seventh Workshop on Topics in Operating Systems*, 1999.
- [2] Daniel Ellard, Michael Mesnier, Eno Thereska, G. R. Ganger, and Margo Seltzer, "Attribute-Based Prediction of File Properties", *Harvard Computer Science Group Technical Report TR-14-03*, December 2003.
- [3] Michael Mesnier, Eno Thereska, Daniel Ellard, Gregory R. Ganger, and Margo Seltzer, "File Classification in Self-* Storage Systems", *Proceedings of the First International Conference on Autonomic Computing (ICAC-04)*, New York, May 2004.
- [4] Carl Staelin and H. Garcia-Molina, "Clustering active disk data to improve disk performance," *Tech. Rep. CS-TR-283-90, Dept. of Computer Science, Princeton Univ., Princeton, N.J.* Sept. 1990.
- [5] Pei Cao, Edward W. Felten, Anna R. Karlin, and Kai Li, "Implementation and Performance of Integrated Application-Controlled File Caching, Prefetching, and Disk Scheduling," *ACM Transactions on Computer Systems*, 14(4):311–343, 1996.
- [6] Daniel Ellard, Jonathan Ledlie, Pia Malkani, and Margo Seltzer. "Passive NFS Tracing of Email and Research Workloads," *In Proceedings of the Second USENIX Conference on File and Storage Technologies (FAST'03)*, pages 203–216, San Francisco, CA, March 2003.
- [7] Daniel Ellard, Jonathan Ledlie, and Margo Seltzer, "The Utility of File Names," *Technical Report TR-05-03, Harvard University Division of Engineering and Applied Sciences*, 2003.
- [8] Gregory R. Ganger and M. Frans Kaashoek, "Embedded Inodes and Explicit Grouping: Exploiting Disk Bandwidth for Small Files," *In USENIX Annual Technical Conference*, pages 1–17, 1997.
- [9] SOS project home page, <http://www.eecs.harvard.edu/sos/index.html>, 2006.